Remote Scripting with IFRAME

by Eric Costello and Apple Developer Connection
02/08/2002

As web sites become more and more like traditional applications, the call-response-reload model used in HTTP transactions becomes increasingly cumbersome. Instead of delivering a single dynamic page, the DHTML or JavaScript developer must create a series of separate pages. The flow of the application is interrupted by page reloads whenever the client communicates with the server. Remote scripting provides a solution to this problem, easing development of complex JavaScript applications, and providing a better experience for the end user.

What is Remote Scripting?

Remote Scripting is the process by which a client-side application running in the browser ar server-side application can exchange data without reloading the page. Remote scripting allo you to create complex DHTML interfaces which interact seamlessly with your server.

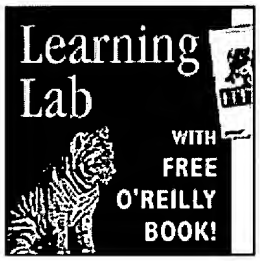
If you're not clear on exactly what this means, think of the ever-present JavaScript image sw (you've coded one of those, haven't you?). In an image swap, your client-side code requests data from the server to be displayed on the web page; in this case the request made to the se for a new image with which you wish to replace an existing image. But what if you could as server for something other than an image? What if you could request a block of text? And w your request could be more than a simple call for data? What if you could submit form data to the server, have the server process that data and respond with an appropriate message? O course, all of these things are already possible by relying on page reloads, but remote scripti allows complex interaction with the server that appears as seamless to the user as a simple i swap.

Remote scripting is a form of RPC (Remote Procedure Call), which is a general term used t describe the exchange of data between remote computer systems. Remote scripting opens u great number of opportunities for the developer. Imagine a news article that can load side b graphical information related to the article directly into the web page when the site's visitors request it. Imagine a gallery of photo thumbnails that turn into full-sized images when click Imagine a full-featured, browser-based content management system interface that allows a s administrator to edit web site copy in situ. The possibilities are limited only by the creativity

- Developer Tutorials
- Computer Hardware
- Search Engine Optimization

Resource Centers
Perl
Java
Python
C/C++
Scripting
Web
Digital Media
Web Services
XML
Oracle
SysAdm/Networking
Security
Databases
Linux/Unix
Macintosh/OS X
Windows
.NET
Open Source
Wireless
Bioinformatics

ATOM FEED
RSS 1.0
RSS Feed
Using our RSS Feeds



Related O'Reilly Books

- ActionScript Cookbook
- ActionScript for Flash MX Pocket Reference
- Amazon Hacks
- Apache Cookbook
- Cascading Style Sheets: The Definitive Guide, 2nd Edition
- Content Syndication with RSS
- Dreamweaver MX 2004: The Missing Manual
- eBay Hacks
- Essential ActionScript 2.0
- Flash Hacks
- Flash Remoting: The Definitive Guide

the developer.

Setting up Remote Scripting

As I'm sure you've gathered, remote scripting is a client/server technology that relies on twc distinct components: the server, which is your web server, and the client, which is the JavaS application in your web page. In remote scripting, you'll be making an RPC from your Java! application to your server. You'll need to be versed in both client and server web technologi get started with remote scripting.

The client-side component is a part of the web page that you want to enable with remote scr capabilities. It needs to be able to make a call back to the server, usually done via HTTP, an then be able to receive and deal with the server response, if any. The server component mus ready to receive requests from the client, process the request, and then respond, returning da the client if needed. The server component can be a script served up by your web server, suc a .php, .asp, or .cgi file, or it can be a dedicated software package that handles only RPC cal it can even be a simple database comprised of static files.

As you can see, the requirements for each of these components will vary greatly depending how you choose to implement your remote scripting system.

Remote Scripting Technology Options

Remote scripting is actually a fairly broad term, and does not necessarily imply the use of a particular technology. Several remote scripting options are available to the web developer:

Java Applet/ActiveX control/Flash By embedding an ActiveX component, a Java applet, Flash movie into your web page, all of which are capable of making HTTP requests and interacting with your client-side JavaScript code, you can implement the client component c remote scripting system. Not all browsers support the technology (called LiveConnect) that for interaction between JavaScript and such objects; most notably, IE5 Mac does not suppor all, and NS6 (on all platforms) has such splotchy support it cannot really be called support a In addition to those problems, using an embedded object for remote scripting requires the er to install additional proprietary software. ActiveX does not work on the Macintosh platform Java can cause some difficulties with some versions of Internet Explorer. On the upside, Jav ActiveX can create socket connections with the server, which allows the server to interact w your application even when communication is not initiated by the client component. Unless developing in an environment where browser homogeneity can be assumed, these technolog may not be a good choice for remote scripting.

XML-RPC is in many ways the superior choice for remote scripting. It uses XML for encapsulating data, HTTP for its transport mechanism, and can be implemented in a wide va of platform and software environments. XML-RPS is an open, standards-based technology quickly becoming the de-facto method for RPC of all kinds. The down side is that it takes a more know-how and work to get it up and running, requiring the installation of XML-RPC libraries on your server and in your client-side code. It also does not work with IE5 on the Macintosh.

Simple HTTP via a hidden IFRAME Using an IFRAME in conjunction with a script on y web server or a database of static HTML files, is by far the easiest of the remote scripting op available. The IFRAME has been part of the HTML specification since version 4 and is sup in nearly all modern browsers. On the server, you can use your scripting language of choice process page requests made to the IFRAME. In the remainder of this article you'll see how t implement these components.

Using an IFRAME for Remote Scripting

Enough already with the talk! Let's get to the code. I'm going to show you everything you n know to set up a remote scripting system with an IFRAME. I'll start with the basic set up, a describe some common browser problems and their solutions.

For starters, you'll need two web pages. The first should be called `client.html` (or some variation thereof, depending on the example). It will be the main working document and wil contain nearly all of the scripts. The `client.html` will be making remote scripting calls usi IFRAME embedded in the document's HTML. The second page will be `server.html`. In a server environment, you'd call it `server.php`, or `server.asp`, or whatever extension fits yc server platform, but in this example we're not using a dynamic server component, so an `.ht` extension is fine.

The `client.html` file will make RPC calls by passing arguments to `server.html` in the qu string. The `server.html` will process the RPC, and write out a very simple page which con script calls back to `client.html`. This will be clearer after we look at a simple example.

In `client.html` enter the following:

```
<script type="text/javascript">
function handleResponse() {
  alert('this function is called from server.html')
}
</script>

<iframe id="RSIFrame"
  name="RSIFrame"
  style="width:0px; height:0px; border: 0px"
  src="blank.html"></iframe>

<a href="server.html" target="RSIFrame">make RPC call</a>
```

As you can see, in it's simplest form IFRAME remote scripting simply points the `TARGET` at of a link to a hidden IFRAME. You can hide an IFRAME by setting its WIDTH, HEIGHT, BORDER properties to 0px. Don't use `display:none`, because NS6 ignores IFRAMEs whe `display` property is set to `none`. If you were using the above code with `display:none` set f IFRAME, the contents of `server.html` would load into the main `client.html` document, into the IFRAME.

In `server.html` use the following HTML:

```
<script type="text/javascript">
  window.parent.handleResponse()
</script>
```

The `handleResponse()` function is found in our `client.html` page, and we simply use the `parent` property of the IFRAME document's `window` object to call it. See it in action here. (course you notice that in this example we are not actually processing anything on the server you want, on your server go ahead and change your server page code to something like this using ASP as an example server scripting language):

```
<script type="text/javascript"<
window.parent.handleResponse('<%=request.servervariables("SERVER_SOFTWARE'
</script>
```

And change the `handleResponse()` function in `client.html` to something like this:

```
<script type="text/javascript">
function handleResponse(msg) {
  alert(msg)
}
</script>
```

You should now get an alert message containing information returned from your server, the results of your first remote procedure call!

Should you want to pass data to `server.html`, you can either embed it in the query string o URL specified in your link, or, to allow user interaction, use a form instead of a link, like sc

```
<form action="server.html" target="RSIFrame">
<input type="text" name="whatever">
</form>
```

Just like the link, you specify the IFRAME as the target of your form, use `server.html` to process the form data on the server, and send any data back with the `handleResponse()` m More on working with forms later.

Problems and Solutions

You may have already realized that there are some serious problems with this simplest of re scripting scenarios. Perhaps most seriously, this method renders the "back" and "reload" but most browsers useless. Because the loading of `server.html` in the IFRAME is added to the browsers `history` object, hitting the reload button after you've loaded `server.html`, for ins reloads `server.html` in the IFRAME instead of reloading `client.html` as would be expec

NOTE: You may or not experience this problem, depending on a combination of factors including your browser version, server platform, HTTP headers, and browser settings. Trust me, if you simply target a link at an IFRAME, some users will have problems with their reload and back buttons.

A new function called `callToServer()` handles this problem by using the `replace()` method of the IFRAME document's `document` object. In addition, instead of creating an IFRAME element in the HTML, I'll let `callToServer()` create the IFRAME through the wonders of the DOM. Doing so alleviates all the reload and back button problems, and keeps the mark up free from unneeded tags.

Sadly, referencing the IFRAME's `document` object is no simple task, since IE5, IE5.5, IE6, and NS6 all provide different ways to access it. IE5, both on the Mac and PC, provides the simplest method: IFRAMEs in this browser show up in the `document.frames` array and have a `document` property. In IE5.5, IE6 and NS6, you can retrieve the IFRAME element object with `document.getElementById()`, then, in NS6, use the object's `contentDocument` property, and in IE 5.5+, use the `document` property of the IFRAME object's `contentWindow` property. That's quite a mouthful, isn't it? Thankfully, it's quite a bit simpler in the

actual code:


```
var IFrameObj; // our IFrame object
function callToServer() {
  if (!document.createElement) {return true};
  var IFrameDoc;
  var URL = 'server.html';
  if (!IFrameObj && document.createElement) {
    // create the IFrame and assign a reference to the
    // object to our global variable IFrameObj.
    // this will only happen the first time
    // callToServer() is called
    var tempIFrame=document.createElement('iframe');
    tempIFrame.setAttribute('id','RSIFrame');
    tempIFrame.style.border='0px';
    tempIFrame.style.width='0px';
    tempIFrame.style.height='0px';
    IFrameObj = document.body.appendChild(tempIFrame);

    if (document.frames) {
      // this is for IE5 Mac, because it will only
      // allow access to the document object
      // of the IFrame if we access it through
      // the document.frames array
      IFrameObj = document.frames['RSIFrame'];
    }
  }

  if (navigator.userAgent.indexOf('Gecko') !=-1
    && !IFrameObj.contentDocument) {
    // we have to give NS6 a fraction of a second
    // to recognize the new IFrame
    setTimeout('callToServer()',10);
    return false;
  }

  if (IFrameObj.contentDocument) {
    // For NS6
    IFrameDoc = IFrameObj.contentDocument;
  } else if (IFrameObj.contentWindow) {
    // For IE5.5 and IE6
    IFrameDoc = IFrameObj.contentWindow.document;
  } else if (IFrameObj.document) {
    // For IE5
    IFrameDoc = IFrameObj.document;
  } else {
    return true;
  }

  IFrameDoc.location.replace(URL);
  return false;
}
```

With that function added to `client.html`, and the IFRAME tag removed, you now can sim add `callToServer()` to the link as an `onclick` event handler that returns `false` if the IFR/ exists and can be loaded with `server.html`, and otherwise returns `true`, allowing the link t function as normal. I've removed the TARGET attribute of the link, and changed the `href` a pointing it to the blank page. You'll probably want to point the link to a page on your server

explains to the user that their browser does not support Advanced IFRAME Remote Scriptii Technology, or better yet, which performs the same tasks as `server.html` and then redirect user back to client.html.

```
<a onclick="return callToServer();" href="blank.html">make RPC call</a>
```

You can see this more complete code in action here.

Using a Form to Pass Data to the Server

Now that you've got the basic framework in place, it would be nice to add a form to send da the server and receive back a useful response. You can't simply target the IFRAME with a f anymore, as you've learned, and you are limited to passing data to `server.html` via the que string. What you need is a function to shuffle values from a form into a properly formatted (string which can then append to the URL passed to the `replace()` method of the IFRAME' `document.location` property.

```
function buildQueryString(theFormName) {
  theForm = document.forms[theFormName];
  var qs = ''
  for (e=0;e<theForm.elements.length;e++) {
    if (theForm.elements[e].name!='') {
      qs+=(qs=='')?'?':'&'
      qs+=theForm.elements[e].name+'='
            +escape(theForm.elements[e].value)
      }
    }
  return qs
}
```

The function `buildQueryString()` takes one parameter, the name of the form from which wish to grab data. It iterates through each element of that form, and for those elements with properties, it adds a new name/value pair to a string called `qs`. The function then returns `qs`. `buildQueryString()` in place, you can now change `callToServer()` to also accept a form as parameter, and then within `callToServer()` we change the line

```
var URL = 'server.html';
```

to

```
var URL = 'server.html'
  + buildQueryString(theFormName);
```

Next, create a form like so:

```
<form name="emailForm" id="emailForm"
  action="server.html"
  onsubmit="return callToServer(this.name)">
Your name:<br>
<input type="text" name="realname"><br>
Your message:<br>
<textarea name="message" cols="50" rows="10">
</textarea>
<br><br>
<input type="submit" value="submit">
</form>
```

Now we have a form that, when submitted, sends its data to the server through a hidden IFR Your server can act on that data however you wish, interacting with client.html by calling `handleResponse()`. Here's an example that sends form data back to the server, hides the fo and displays a success message.

Looking at the code, you'll see that I changed the `handleResponse()` function in the client so that instead of popping up an alert message it now takes care of hiding and showing the relevant page elements.

A Final Example

To help you see the possibilities that Remote Scripting affords you the developer, I've work one final example that loads data from a database of static HTML files. You can view that example here.

That should give you a taste for what can be done with RPC. Once you start playing around on your server, you'll really get a feel for the power in your hands. Try making calls to your to retrieve data from a database, passing the data back to your client page with a call to the `handleResponse()` function, and then adding that data to your client page using DOM met like those detailed in these Internet Developer articles: DOM-2 Part 1 and DOM-2 Part 2. B to use your new skills for good and not evil!

Further Reading and Outside Resources

If you'd like to explore Remote Scripting further, let me recommend these sites to you:

- Brent Ashley's Remote Scripting Resources, which contains Brent's own RSLite and Javascript Remote Scripting (JSRS) tools along with links to other Remote Scripting resources.
- Virtual Cowboys' vcXMLRPC Library, a very nice implementation of XML-RPC in javascript.

Eric Costello is a contract developer for hire, working out of his mysterious company Schw maintains a personal site at glish.com, where he links to articles on Web standards, DHTM CSS, XML, and other topics of interest to web developers.

Return to the JavaScript and CSS DevCenter.

As remote scripting is used in a variety of environments, some interesting issues are t to arise. Have you encounted anything?

You must be logged in to the O'Reilly Network to post a talkback.

Post Comment Main Thread Only Old

Showing messages 1 through 13 of 13.

Trackback from The realm of Davros
Remote Scripting for Web-Applications
2004-02-08 11:48:26 [Reply | View]

How to retrieve a HTTP response using IFrames?
2003-07-10 16:00:24 anonymous [Reply | View]

Hi,

I'm using a Form and an IFrame that is used by the client to submit a Post request to a server. The target of the Form is the IFrame name. The IFrame has an "onload" event h which I use to retrieve the response from the server. The response is pure text (no HTM and consists of a string that is formatted based on a protocol. My question is: How do I access to the reponse string from within my "onload" event handler? I tried using the following but don't get anything:

var resp = document.getElementById("frameID").innerHTML;

How do I solve this problem? Any help would be greatly appreciated! Thanks!

A cleaner solution
2003-05-11 21:50:17 anonymous [Reply | View]

Here is a simpler solution:
http://www-106.ibm.com/developerworks/web/library/wa-exrel/?dwzone=web

It doesn't use query strings, but javascript variables.

I have a problem with a flashing statusbar/curor, as mentioned in a previous post. Does someone have a solution to this?

Best Regards
Stefan Krabbe

A cleaner solution
2003-05-18 11:56:49 anonymous [Reply | View]

Yes it's me again. I solved the problem with the blinking cursor and status bar, an now exchange information with the web server without reloading the HTML page without any flicker.

This example uses a small javaapplet instead of the IFRAME.

The example contains 4 files, that enables a html page to exchange information w web server without reloading the viewed HTML page (client.php).

The solution uses javascript, java and PHP. You will propably need to add some database too.
I've used PHP in these examples, but it should be easy to replace the PHP with AS JSP, or the like.

Check out this page for the other method, that does the same job:
http://www-106.ibm.com/developerworks/web/library/wa-exrel/?dwzone=web
It does not use java, but an IFRAME. It causes the status bar to flicker, and the cu turn into an hour-glass, every time receiver.php gets reloaded.

My solutions solves this annoyance, by loading the file from a java applet, by ope

socket back to the webserver.

Here is a short description of the files:
--client.php--
This file contains the webpage that needs to get information from the webserver w
getting reloaded
This is propably the page the user sees when using your service.

--receiver.php--
This is the webpage that we load, to avoid loading client.php.
The user never sees this page.

--ClientApplet.java--
This is java program we use to load reciever.php behind the scenes.
The user never sees this applet.

--Client.java--
This file contains the methods that gets/reloads/refreshes reciever.php from the webserver.
The user never sees this applet.

Best Regards
Stefan Krabbe
dsl74896@vip.cybercity.dk

PS: The 4 files follows this line.
=- client.php -=-=-=-

```
<param name="host" value="www.somesite.somedomain" />
<param name="port" value="80" />

Your browser is not able to run Java applets. This is probably due to having Java
disabled in your browser settings.


</applet>
</body></html>

=- receiver.php -=-=-=-
```

```
private Socket socket;
private DataOutputStream toserver;
private DataInputStream fromserver;
private byte received_data[];
// Applet parameters
private String host;
private int port;
public Client(String arg_host, int arg_port) {
host = arg_host;
port = arg_port;
received_data = new byte[10000];
fakeid = 0;
}
private static final String extract_javascript(String html) {
String s_begin, s_end;
int i_begin, i_end;
s_begin = "<SCRIPT LANGUAGE=\"JAVASCRIPT\">";
s_end = "</SCRIPT>";
i_begin = html.indexOf(s_begin);
if(i_begin < 0)
return "";
i_begin += s_begin.length();
i_end = html.indexOf(s_end, i_begin);
if(i_end < 0)
return "";
return html.substring(i_begin, i_end);
}
public final String myprocess(String querystring) {
String received = "";
try {
// Use the querystring to pass information to receiver.php
String msg = "GET /receiver.php?fakeid" + String.valueOf(++fakeid) + querystri
HTTP/1.1\r\n";
msg += "Host: " + host + "\r\n";
msg += "Connection: close\r\n";
//msg += "Cookie: PHPSESSID=" + somesessionid + "\r\n"; // Set this if you nee
pass session information
msg += "\r\n";
socket = new Socket( host, port );
fromserver = new DataInputStream( socket.getInputStream() );
toserver = new DataOutputStream( socket.getOutputStream() );
toserver.writeBytes(msg);
int n = socket.getSoTimeout();
boolean bad = (socket.isClosed() || !socket.isConnected() || socket.isInputShutdow
socket.isOutputShutdown());
int numreads = 0;
for(received = ""; !bad; ) {
int bytes_received = 0;
bytes_received = fromserver.read(received_data, 0, 9990);
if(bytes_received <= 0)
break;
received += new String(received_data, 0, bytes_received);
bad = (socket.isClosed() || !socket.isConnected() || socket.isInputShutdown() ||
socket.isOutputShutdown());
```

```
++numreads;
}
received = extract_javascript(received);
fromserver.close();
toserver.close();
socket.close();
} catch( IOException ioerr ) {
System.out.println("Error: " + ioerr.toString());
return "";
}
return received;
}
}
```

limitations
2003-01-30 16:34:44 anonymous [Reply | View]

Using an xp machine, i have encountered a request.querystring limitation. my machine not allow request.querystring to pass data for more than 1500 characters. As such, the callToServer does not process. it is a bit odd considering that my win 2000 server does burb on this bug.

limitations
2003-04-22 08:12:52 anonymous [Reply | View]

Same limitations under IE6.
I haven't determined exact number of characters at which this "kicks in", but I cou send a 3000 character text using IE6 and Windows NT. Limitation does not apply 7 on same hardware.

Hmmm. I'm goint to experiment with adding a form to the IFRAME or for that m the document itself, then do a hidden post of that field.(or fields). If it works I'll p here.

Serge d'Adesky sdadesky@akulink.com

Start Navigation sound and cursor during RPC call
2002-12-09 19:41:52 anonymous [Reply | View]

I have developed a system that polls the server every second using this RPC technique. works great, but the "start navigation" sound, a "click" on my PC, sound every time I d Also the cursor changes from an arrow to an arrow with an hourglass. Then there is the of the server procedure flashing by in the status bar.

This is a visual and auditory distraction. While I can get rid of the noise and the status the cursor changing once a second is VERY annoying.

Right now my only real solution is to poll less frequently, say every 10 seconds. Is the better solution?

w
2002-09-19 18:55:33 anonymous [Reply | View]

w

RE: IE5.0 error - a solution
2002-07-08 09:25:55 sanjev [Reply | View]

Thanks for the Article Mr. Costello.

This is mostly a reply to Geeet's error issue, however. Some time ago when writing coc
do something along the lines of what you are doing I encountered the same problem. Y
will recieve "Class doesn't support Automation: on the:
IFrameObj = document.frames['RSIFrame'];
OR (if you avoid that)
you will find that:
IFrameObj.document === document
(Clearly you do not wish to write to that!)

One solution is to dowload the iframe in the main page (form the server) and then man
it (which works on IE, but not NS6, your your code gets ugly) but that is undesirable (t
have been doing it)

A better solution is, instead of:
var tempIFrame=document.createElement('iframe');
...
IFrameObj = document.body.appendChild(tempIFrame);

try:
var dv = document.createElement ('div');
dv = document.body.appendChild (dv);
dv.innerHTML = '<iframe id="RSIFrame"></iframe>'
var iframe = document.getElementById("RSIFrame")

then the document.frames["RSIFrame"] will succeed along with all code that follows. (
that is ONLY they way to do it on IE5.0, all other browsers can user the *much* clean
code provided in this artice.

ie 5.0 windows javascript error
2002-03-04 08:13:07 geeeet [Reply | View]

The example that submits the form doesn't work in ie5.0 on windows.. unfortunaly i di
find out until i completed at site using this.. :(

Anyone found solutions for ie5.0 windoze ?

/geeeet@ghtml.com

ie 5.0 windows javascript error
2004-01-07 07:15:02 anonymous [Reply | View]

Try http://developer.apple.com/internet/javascript/iframe.html

k>to)

remote scripting vs SOAP
2002-02-16 11:09:57 divad [Reply | View]

It seems to me that remote scripting is a powerful and readily available technique for building distributed applications. So why would one go to the trouble of using SOAP v you can simply use remote scripting? Am I alone in this view?

remote scripting vs SOAP
2002-10-28 12:02:18 inetws [Reply | View]

SOAP gives you much more power and flexibility since it can be integrated into a fledged program. A very simple example would be an address book program on y computer could contact a zip code address soap service to automatically fill in cit information once you put in a zip code.